AUTOMATIC BIOS RECOVERY IN A MULTI-NODE COMPUTER SYSTEM

Inventors: Fang Lu
13219 Vendrell Drive
Austin, Texas 78729

Frank L. Wu
8805 Tweed Berwick Drive
Austin, Texas 78750

Assignee: DELL PRODUCTS L.P.
One Dell Way
Round Rock, Texas 78682-2244

BAKER BOTTS L.L.P.
One Shell Plaza
910 Louisiana
Houston, Texas 77002-4995

Attorney Docket: 016295.0749
(DC-03285)

AUTOMATIC BIOS RECOVERY IN A MULTI-NODE COMPUTER SYSTEM

TECHNICAL FIELD

The present disclosure relates in general to information handling systems. In particular, this disclosure relates to systems, methods, and program products for recovering from basic input/output system (BIOS) problems such as BIOS image corruption in multi-node computer systems.

BACKGROUND

As the value and use of information continues to increase, individuals and businesses seek additional ways to process and store information. One option available to users is information handling systems. An information handling system generally processes, compiles, stores, and/or communicates information or data for business, personal, or other purposes thereby allowing users to take advantage of the value of the information. Because technology and information handling needs and requirements vary between different users or applications, information handling systems may also vary regarding what information is handled, how the information is handled, how much information is processed, stored, or communicated, and how quickly and efficiently the information may be processed, stored, or communicated. The variations in information handling systems allow for information handling systems to be general or configured for a specific user or specific use such as financial transaction processing, airline reservations, enterprise data storage, or global communications. In addition, information handling systems may include a variety of hardware and software components that may be configured to process, store, and communicate information and may include one or more computer systems, data storage systems, and networking systems.

One type of information handling system currently involved in intense development efforts is the multi-node computer system (MCS). For instance, in one proposed configuration, the 82870 chipset developed by Intel

Corporation (hereinafter the "870 chipset") provides for a four-node system that includes a set of central processing units (CPUs), a CPU hub, an input/output (I/O) hub, and a PCI expansion bus for each node. The CPU hubs may also be called scalable node controllers (SNCs). In addition, the 870 chipset features a multi-port switch, known as a scalability port switch (SPS), in each hub, and the SPS may be configured to interconnect an I/O hub from one node to the CPU hub of another node, for example. SPS configuration is handled by the CPU hubs.

In an MCS, as in more traditional computer systems, a basic input/output system (BIOS) may be used for tasks such as system initialization. However, a problem may be presented when BIOS code in a computer system becomes corrupted, for example as the result of a hardware malfunction.

SUMMARY

The present disclosure relates to a system, a method, and software for automatically recovering from BIOS corruption. In one example embodiment, a multi-node computer system (MCS) features first and second nodes containing first and second sets of central processing units (CPUs), respectively. The first and second nodes also include first and second firmware units, respectively, and each firmware unit contains BIOS code. In addition, the MCS includes BIOS recovery logic, in at least one of the first and second firmware units, that automatically recovers from BIOS corruption. For example, the BIOS recovery logic may automatically detect corruption in the BIOS code in the first node and, in response, automatically replace the corrupt BIOS code with good BIOS code from the second node.

By contrast, in prior systems, when the BIOS gets corrupted, the typical recovery process is a manual, time-consuming process. For instance, the user may be required to perform the following manual steps:

(1) determine that the BIOS has malfunctioned,
(2) power off the system,
(3) open the chassis,
(4) set a jumper,
(5) insert into a disk drive a recovery media such as a floppy disk with a clean BIOS,
(6) power on the system to start recovery,
(7) verify successful BIOS recovery,
(8) reboot the system, and
(9) reassemble the chassis.

According to the process of the present disclosure, the BIOS may be recovered without any manual intervention.

Various features of one or more embodiments of the present invention are described at some length below, with reference to one or more example implementations. Upon review of this disclosure, numerous additional advantages and alternative embodiments of the invention will be readily apparent to those of ordinary skill in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

The present disclosure and its numerous objects, features, and advantages may be more fully understood by reference to the following description of an example embodiment and the accompanying drawings, in which:

FIGURE 1 presents a block diagram of an example embodiment of a multi-node computer system with support for automatic BIOS recovery;

FIGURE 2 presents a flowchart of an example embodiment of a process for automatically performing BIOS recovery;

FIGURES 3-4 present example memory maps for two firmware units from FIGURE 1; and

FIGURE 5 presents an example memory map for the multi-node computer system of FIGURE 1.

DETAILED DESCRIPTION OF AN EXAMPLE EMBODIMENT

For purposes of this disclosure, an information handling system may include any instrumentality or aggregate of instrumentalities operable to compute, classify, process, transmit, receive, retrieve, originate, switch, store, display, manifest, detect, record, reproduce, handle, or utilize any form of information, intelligence, or data for business, scientific, control, or other purposes. For example, an information handling system may be a personal computer, a network storage device, or any other suitable device and may vary in size, shape, performance, functionality, and price. The information handling system may include random access memory (RAM), one or more processing resources such as a central processing unit (CPU) or hardware or software control logic, read only memory (ROM), and/or other types of nonvolatile memory. Additional components of the information handling system may include one or more disk drives, one or more network ports for communicating with external devices, and various input and output (I/O) devices, such as a keyboard, a mouse, and a video display. The information handling system may also include one or more buses operable to transmit communications between the various hardware components.

Referring now to FIGURE 1, there is illustrated an information handling system 10 that includes two or more nodes 15 and 20, with each node containing its own set of processors and I/O. Information handling system 10 may also be called a multi-node computer system (MCS) 10. In the illustrated embodiment, MCS 10 is implemented using a

chipset, such as the 870 chipset, that allows the nodes to share processing resources, such as CPUs, and to share I/O resources, such as connections to disk drives or other storage or I/O devices. Consequently, MCS 10 enjoys increased computing power and I/O flexibility, and MCS 10 provides increased capabilities for parallel processing and storage, network, and data I/O, relative to previous architectures.

Specifically, in the example embodiment, node 15 includes one or more CPUs 25a-25d coupled to a CPU hub 35. CPU hub 35 provides access to a CPU firmware unit 45 and to RAM 55. In the example embodiment, CPU firmware unit 45 contains a system BIOS 47a for initializing components of node 15 such as CPUs 25a-25d, RAM 55, and I/O. CPU firmware unit 45 may also be called a north firmware hub (FWH) 45. Similarly, node 20 includes a set of one or more CPUs 30a-30d, a CPU firmware unit 50 containing BIOS code 52a, and RAM 60, all interconnected by a CPU hub 40. In the example embodiment, CPU hubs 35 and 40 are implemented as scalable node controllers (SNCs) 35 and 40, and SNCs 35 and 40 serve as memory controllers and host bridges for their respective nodes. For example, some of the functions performed by SNC 35 may be functions traditionally performed by a north bridge.

Furthermore, in addition to the connections described above, SNC 35 connects to two multi-port switches for internodal communications 65 and 70. One of those multi-port switches 65 resides within node 15, and the other multi-port switch 70 resides in node 20. Similarly, CPU hub 40 connects to multi-port switch 65

and to multi-port switch 70. In the example embodiment, multi-port switches 65 and 70 are implemented as scalability port switches (SPSs) 65 and 70. Thus, the SNC in each node is operable to provide a connection with the SPS of the other node.

Node 15 also includes an I/O hub 75, which is connected to SPS 65. In addition, I/O hub 75 is connected to SPS 70 from node 20. Likewise, node 20 includes an I/O hub 80 that is connected to SPS 70 and to SPS 65 in node 15. I/O hubs 75 and 80 may also be called south I/O hubs (SIOHs) 75 and 80.

A peripheral component interconnect (PCI) hub 95 and an I/O FWH 87 are connected to SIOH 75, possibly via an I/O controller hub (ICH) 85. PCI hub 95 is connected to one or more PCI slots 105, which enable adapter cards to be added to MCS 10. Similarly, in node 20, I/O hub 80 is connected (possibly via an ICH 90) to SPS 70, to an I/O FWH 92, and to a PCI hub 100 with access to one or more PCI slots 110. I/O FWHs 87 and 92 may also be called South FWHs 87 and 92.

According to the example embodiment, South FWH 87 includes a South BIOS image 89, as well as a spare copy 47b of system BIOS 47a. Likewise, South FWH 92 includes a South BIOS image 94, as well as a spare copy 52b of the system BIOS. As explained in greater detail below, spare copies 47b and 52b may be used to recover from BIOS errors in system BIOS 47a and 52a. South BIOS images 89 and 94 may include BIOS code for adapter ROMs and extended system configuration data (ESCD), for example. The various firmware units/hubs store data such as the BIOS images using any suitable storage components,

including nonvolatile memory technologies such as read only memory (ROM), electrically erasable programmable read only memory (EEPROM), flash memory, etc.

Referring now to FIGURE 2, a flowchart illustrates an example embodiment of a process for automatically recovering from BIOS errors such as corruption. The illustrated process involves nodes 15 and 20 in MCS 10. However, in alternative embodiments, a similar process could be used to perform BIOS recovery in multi-node computer systems with more than two nodes. The example process begins with MCS 10 starting an initialization process, for example in response to being powered up or reset. Embedded service management (ESM) software in MCS 10 then uses a left-to-right sequence to select a start node (e.g., node 15) and a bootstrap processor (BSP) (e.g., CPU 25a) within the selected boot node, as shown at blocks 200 and 210. As depicted in FIGURE 1, each node may include an ESM software module 125 or 130, for example. ESM modules 125 and 130 may be implemented, for example, as firmware, as hardwired hardware modules, as configurable hardware modules such as field programmable devices (FPDs), etc. As shown at block 212, MCS 10 then clears a BIOS checksum flag for the current node (i.e., MCS 10 sets that flag to zero). For example, the checksum flag may be stored in a nonvolatile scratch pad memory 37, in SNC 35 (or, for node 20, a scratch pad 42). At block 214, MCS 10 then retrieves the BIOS recovery code from North FWH 45 and begins executing that code.

For instance, FIGURE 3 depicts a memory map for North FWH 45, and that map show that, in the current implementation, BIOS image 47a occupies 16 blocks of

storage. A particular one of those blocks (in this case, block 15) may contain the error recovery code 300. Error recovery code 300 may also be called "minimum capability code" or simply "recovery code." Also, North FWH 50 may contain substantially the same BIOS image as North FWH 45.

At block 216 of FIGURE 2, error recovery code 300 uses SNC 35 to open ports in SPS 65 to other nodes, such as node 20. Specifically, error recovery code 300 configures SNC 35 and SPS 65 in a way that allows other nodes to communicate with node 15 via SPS 65. This configuration process may be called strapping the minimum hardware path.

At block 218, error recovery code 300 then checks system BIOS 47a for problems, for instance by computing a checksum to determine whether system BIOS 47a has been corrupted. As shown at block 222, if system BIOS 47a is corrupt, error recovery code 300 checks for the presence of South FWH 87. For instance, FIGURE 5 depicts an example memory map for MCS 10, and a region in the uppermost portion of that memory space may be reserved for hardware management purposes. Specifically, in the example embodiment, the region reserved for hardware management purposes may be a 32 megabyte (MB) region right before the 4 gigabyte (GB) mark. As illustrated, the 4 MB region right below 4 GB may be called the local firmware region, and that region may provide a window into the North FWH of the current node. Also, the next 12 MB may be known as the global firmware region, and that region may provide a window into the South FWH of the current node.

If error recovery code 300 detects South FWH 87, error recovery code 300 then determines whether the checksum for the contents of South FWH 87 are good, as shown at block 224. At block 226, if the checksum is good, error recovery code 300 copies the spare copy 47b of the system BIOS from South FWH 87 to North FWH 45 to recover system BIOS 47a.

However, if the checksum for spare BIOS 47b is not good, or if South FWH 87 is not detected, the process passes to block 230, and a BIOS checksum flag is set for the current node. At block 232 error recovery code 300 configures a portion of RAM 55 to make that portion ready to store and return data. For instance, with reference to FIGURE 5, error recovery code 300 may initialize a 1 MB portion of RAM starting at a particular address in the system memory block. At blocks 234 and 236, error recovery code 300 then signals the next node in the left-to-right sequence to start and enters a loop waiting for the checksum flag for the current node to get cleared.

As indicated by connector A, which connects block 234 to block 200, when the next node is started, the above process is then initiated in the new node. The new node would then clear its checksum flag, fetch recovery code, etc. If the checksum for system BIOS 47a in the new node is good, the process will eventually pass from block 200 to block 244. Similarly, if the first node has a good BIOS image in North FWH 45, process passes from block 200 to block 244.

Error recovery code 300 then tests whether the checksum flag for the left node (if any) is set. For example, if node 15 had a bad system BIOS image in North

FWH 45, and a bad system BIOS image in South FWH 87 or without South FWH 87, the checksum flag for node 15 would be set at block 230. If node 20 then had a good BIOS image in North FWH 50 or South FWH 92, when node 20 gets to block 244, it will test the checksum flag for node 15. If additional nodes were provided, those nodes would likewise test the flag for the previous (left) node. In the example embodiment, for each node that has started the initiation process, that node will have also opened its SPS ports to other nodes (see block 216). The current node may use those paths to check the flags in the SNCs of the other nodes.

If the checksum flag in the left node is set, at block 246 the current node copies the system BIOS from the current North FWH to the region of memory in the left node that was configured at block 232. At block 248 the current node then release the left node from its wait loop (see block 236) by clearing the checksum flag for the left node. Error recovery code 300 for the current node then configures all of the local RAM at block 249. However, if the determination at block 244 is negative, the current node bypasses blocks 246 and 248 and then configures its RAM at block 249.

At block 250 the current node then tests whether the node to the right is accessible. If it is not, the current node signals the right node to start, and the right node begins processing at block 200, as indicated by block 252 and connector A. The current node then determines whether all nodes are synchronized at block 254. For instance, each node may set a synchronization flag when that node has completed initialization, and

error recovery code 300 may check those flags, or error recovery code 300 may simply wait a predetermined interval to give each node enough time to recover if necessary.

Once all of the nodes are synchronized, a system bootstrap processor (SBSP) is selected at block 256. At block 258 the other processors in MCS 10 are then marked as application processors (APs). At blocks 260 and 262 the APs are halted and the SBSP is started. The SBSP may continue operations, for instance by performing power on self test (POST) instructions, configuring the RAM from all of the nodes as one logical memory structure, etc.

Referring again to block 236, when a node is released from its wait loop, that node then copies the BIOS image from system memory to the North FWH, as indicated at block 238. Of course, before a node is released from its wait loop, another node will have loaded a BIOS image into the system memory for the node being released, as indicated at block 246. After the current node loads the good BIOS image to the North FWH, the current node checks for a South FWH and, if one is found, loads the good BIOS to the South FWH as well, as depicted at blocks 240 and 242. The process for the current node then reaches block 244 and proceeds as described above.

In one alternative embodiment, the recovery code could steps through each node, checking the North FWHs for a good copy of the BIOS, and attempting to recover from the South FWHs only after determining that the system BIOS images in each of the North FWHs are corrupt. Consequently, the sequence of scanning for a good BIOS

image need not necessarily step directly to the I/O module of a node after a corrupted BIOS is found in the CPU module for that node. Instead, the scan sequence could step directly from one CPU module to another upon detection of corruption.

In addition, the recovery code could log recovery information about detected BIOS problems and attempted recovery operations during the recovery process. For instance, the recovery information may be stored in scratch pad 37 or scratch pad 42. Consequently, although the recovery process may be automatic, the logged information may nevertheless allow system administrators to track BIOS errors.

With reference now to FIGURE 4, an example memory map for South FWH 87 shows that I/O data 89 (such as ESCD data) occupies one portion of South FWH 87, while the spare copy 47b of the system BIOS resides in another portion of South FWH 87. South FWH 92 may have the same or similar contents as South FWH 87.

In conclusion, as will be evident from the above description, the recovery process is system self-directed, and requires no user interaction. Since each node with a corrupted BIOS configures itself to allow communication from other nodes (e.g., by setting the SNC, the SPS, and a portion of the memory), once a good BIOS image is found, that BIOS image may be distributed to all of the nodes that need BIOS recovery. For instance, the good BIOS image may cascade in sequence from right to left through the nodes with corrupted BIOS images. In different implementations, the distribution details could vary. For example, the MCS could recover one node at a

time and then reboot the system to recover that next node. Alternatively, the BIOS for all nodes could be recovered at once before the system is rebooted. Since the hardware path from the SNC/SPS down to the firmware hub at the I/O module may be configured as a result of hardware strapping, access to the South FWH may always be guaranteed. Each node may test its South FWH for a good copy of the System BIOS, throughout all the South FWHs until a good BIOS copy is found. The good BIOS may then be distributed to all the nodes that need BIOS recovery. Again, the distribution can be different depending on the implementation.

Since the MCS has at least two nodes, there are always at least two copies of the BIOS image. Furthermore, when considering the South FWHs, additional copies of the system BIOS may also be available. The recovery process is therefore highly reliable. Another advantage of the recovery process is that, since there are already multiple copies of the system BIOS image in the MCS, there is no need to have a dedicated firmware module simply to hold a copy of the system BIOS for recovery purposes only. Additionally, when implemented for chipsets such as the 870 chipset that support both IA-32 and IA-64 systems, the method may perform automatically BIOS recovery for both IA-32 and IA-64 instruction-set architectures.

Although the present invention has been described with reference to one or more example embodiments, those with ordinary skill in the art will understand that numerous variations of the illustrated embodiments could be practiced without departing from the scope and spirit

of the present invention. For example, MCS 10 includes two substantially symmetrical nodes 15 and 20. In some alternative embodiments, two or more nodes could include respective sets of CPUs, and CPU hubs; however, those nodes might share a common I/O hub and a common south firmware unit. The process described above may easily be altered to provide for recovery in that type of MCS or for other types of MCSs.

It should also be understood that the hardware and software components depicted in the example embodiment represent functional elements that are reasonably self-contained so that each can be designed, constructed, or updated substantially independently of the others. In alternative embodiments, however, it should be understood that many of the components, including the recovery code, may be implemented as hardware, software, or combinations of hardware and software for providing the functionality described and illustrated herein. The recovery code may also be called BIOS recovery control logic, to denote various implementations, including software instructions, hardwired logic circuitry, etc.

Alternative embodiments of the invention also include computer-usable media encoding computer instructions for performing the operations of the invention. Such computer-usable media may include, without limitation, storage media such as floppy disks, hard disks, CD-ROMs, read-only memory, and random access memory; as well as communications media such wires, optical fibers, microwaves, radio waves, and other electromagnetic or optical carriers. The recovery

instructions may also be referred to as a program product.

Many other aspects of the illustrated embodiments may also be changed in alternative embodiments without departing from the scope and spirit of the invention. The scope of the invention is therefore not limited to the particulars of the illustrated embodiments or implementations but is defined by the appended claims.